100th Anniversary of
Chula Engineering 2013

4 Repetition

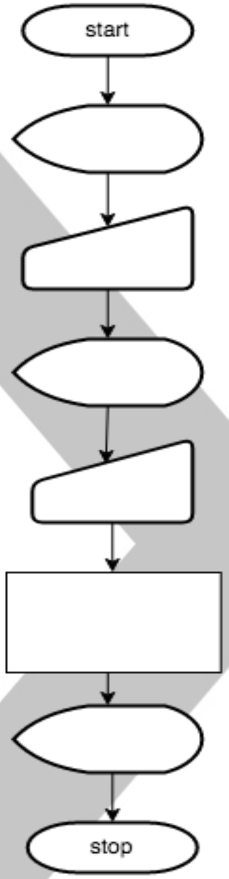
python powered

Topics

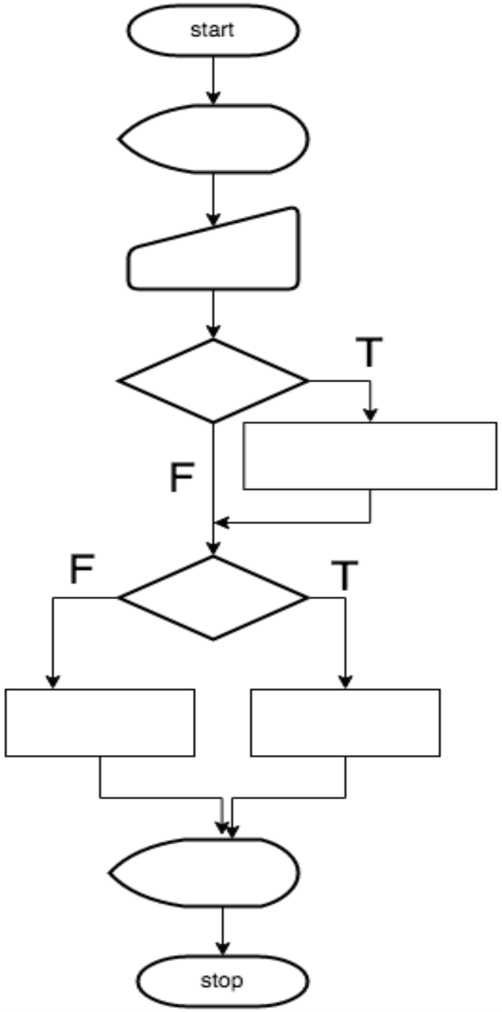
- while
- for
  - for k in range(...)
  - for ch in string
  - for elem in a list
- break

Flowchart

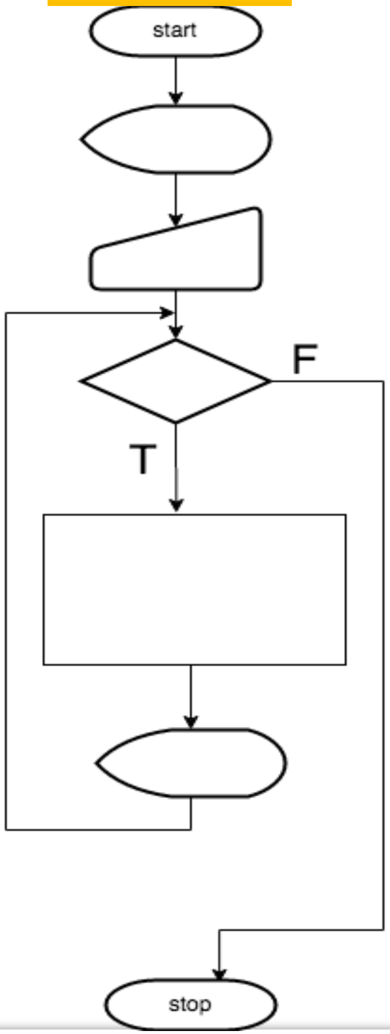
Sequential



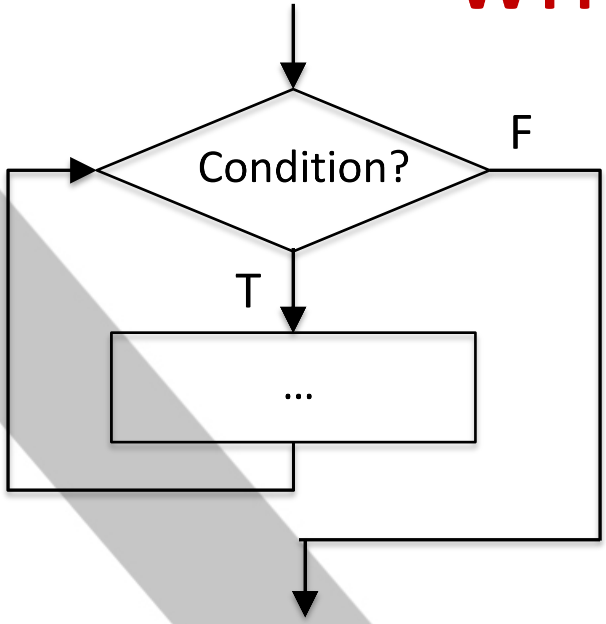
Selection



Repetition



while loop



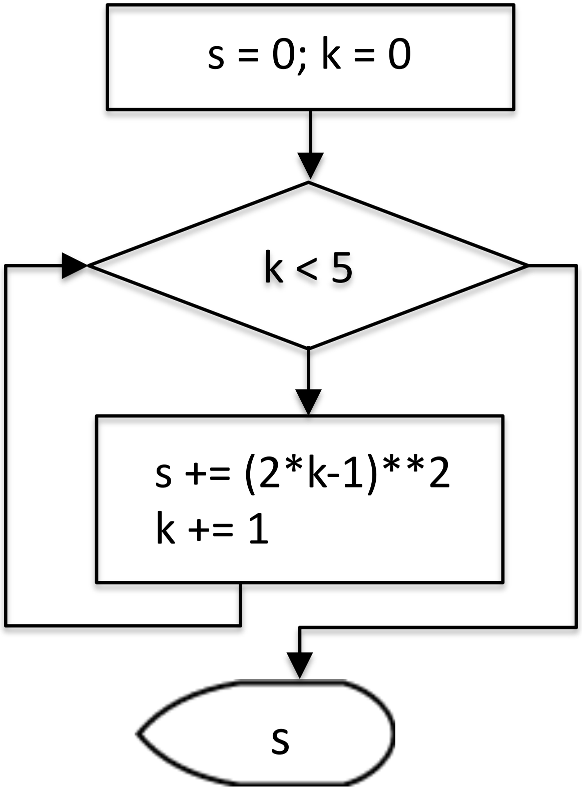
Must end with a colon.

```
while condition :
    instruction set
```

All instructions need to be indented equally from the left.

Example

$$\sum_{k=0}^{4}(2k-1)^2$$



Example

$$\sum_{k=0}^{4} (2k-1)^2$$

(1)

```
s = 0
s += (2*0 - 1)**2
s += (2*1 - 1)**2
s += (2*2 - 1)**2
s += (2*3 - 1)**2
s += (2*4 - 1)**2
print(s)
```

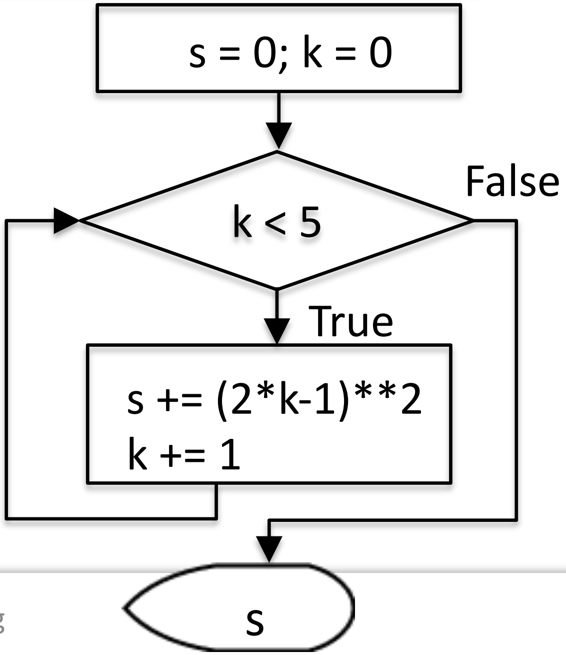
(2)

```
s = 0; k = 0
s += (2*k - 1)**2; k += 1
s += (2*k - 1)**2; k += 1
s += (2*k - 1)**2; k += 1
s += (2*k - 1)**2; k += 1
s += (2*k - 1)**2; k += 1
print(s)
```

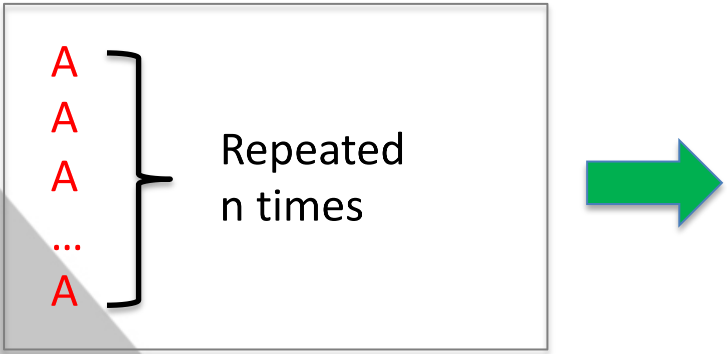
(3)

```
s = 0; k = 0
while k < 5 :
    s += (2*k -1) **2
    k += 1
print(s)
```

Iterate 5 times, k starts from 0, incremented by 1 for each iteration, and terminated when k is 5, as the condition is false.



Loop pattern



A
A
A
…
A

Repeated
n times

```
k = 0
while k < n :
    A
    k += 1
```

OR

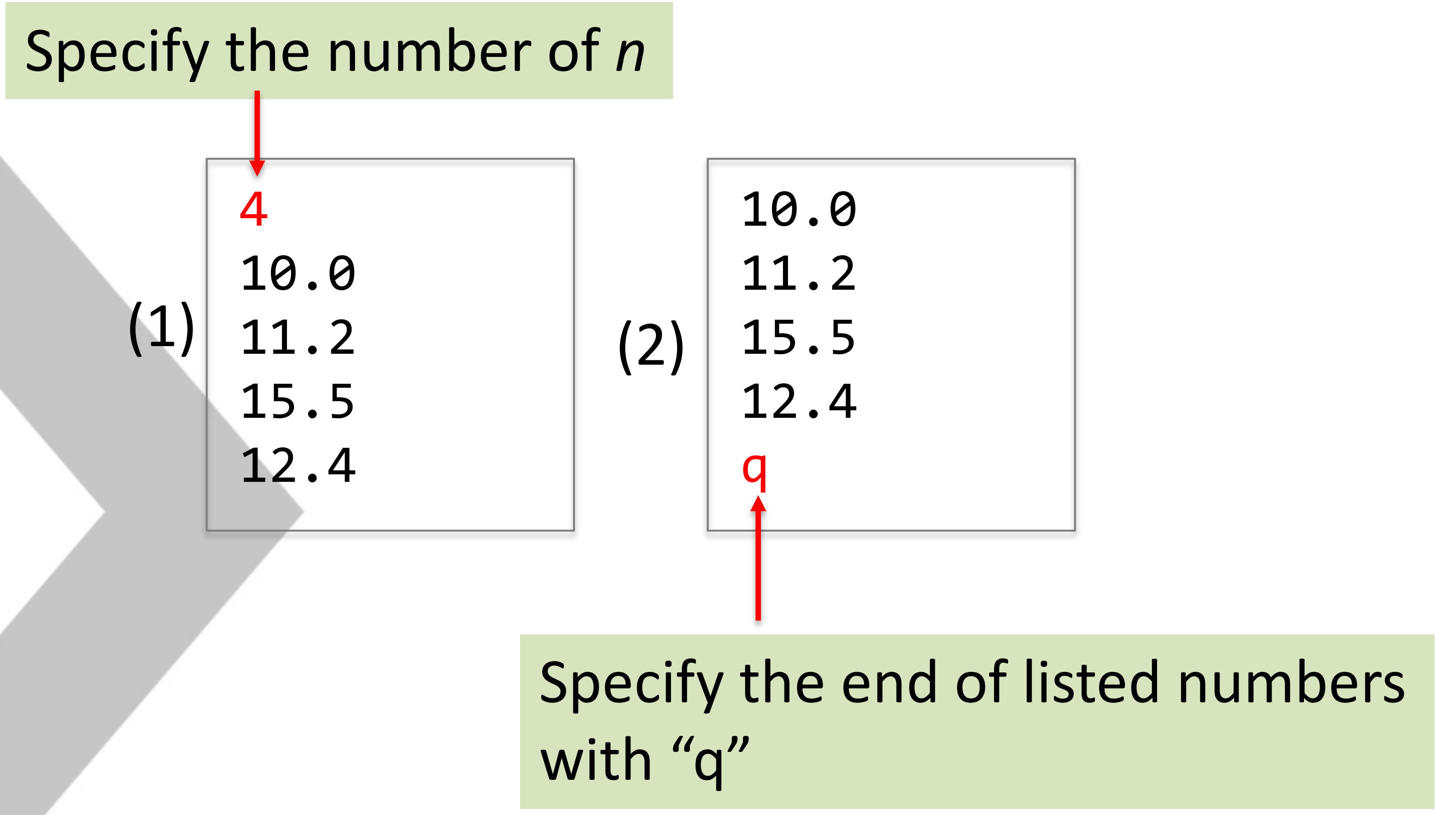
```
k = 1
while k <= n :
    A
    k += 1
```

Example: find the smallest among five numbers

```
min_v = float(input())
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
v = float(input())
if v < min_v:
    min_v = v
print("min = ", min_v)
```

```
min_v = float(input())
k = 0
while k < 4 :
    v = float(input())
    if v < min_v:
        min_v = v
    k += 1
print(" min = ", min_v)
```

Example: find a smallest among *n* numbers



Specify the number of *n*

(1)

```
4
10.0
11.2
15.5
12.4
```

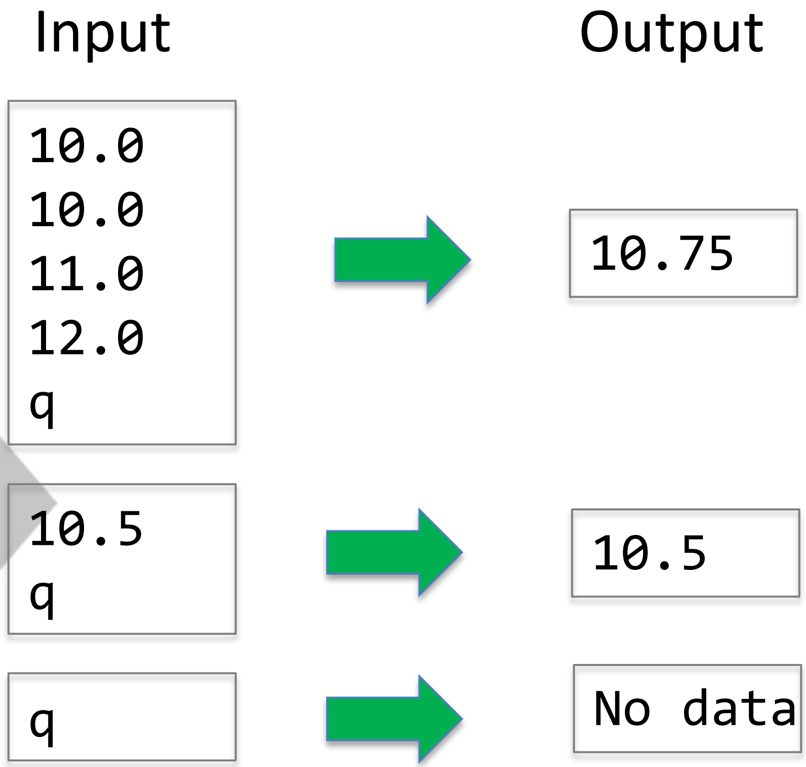
(2)

```
10.0
11.2
15.5
12.4
q
```

Specify the end of listed numbers with “q”

Either (1) or (2) can be the input of the program.

Practice: find the average among *n* numbers



Input

Output

```
10.0
10.0
11.0
12.0
q
```

```
10.75
```

```
10.5
q
```

```
10.5
```

```
q
```

```
No data
```

Example: find a square root using bisection method

- Find a square root of a
- Let L = 0, U = a
- The answer will be within [L, U]
- x = the middle point of L and U
- Repeat the following steps if $x^2 \neq a$
  - If $x^2 > a$:
    - Update the range to [L, x]
  - If $x^2 < a$:
    - Update the range to [x, U]
  - x = the middle point of L and U

$$x = \frac{L+U}{2}$$

a = 25		
L	U	X
0	25	12.5
0	12.5	6.25
0	6.25	3.125
3.125	6.25	4.6875
4.6875	6.25	5.46875
4.6875	5.46875	5.078125
4.6875	5.078125	4.882813
4.882813	5.078125	4.980469
4.980469	5.078125	5.029297
4.980469	5.029297	5.004883

Example: find a square root using bisection method

- Find a square root of a
- Let L = 0, U = a
- The answer will be within [L, U]
- x = the middle point of L and U
- Repeat the following steps if $x^2 \neq a$
  - If $x^2 > a$:
    - Update the range to [L, x]
  - If $x^2 < a$:
    - Update the range to [x, U]
  - x = the middle point of L and U

```
a = float(input())
L = 0; U = a
x = (L + U)/2
while x**2 != a:
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U)/2

print(x)
```

Example: find a square root using bisection method

Got a problem!!

```
a = float(input())
L = 0; U = a
x = (L + U)/2
while x**2 != a:
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U)/2

print(x)
```

```
a = float(input())
L = 0; U = a
x = (L + U)/2
while not close enough:
    if x**2 > a:
        U = x
    else:
        L = x
    x = (L + U)/2

print(x)
```

a and b are not close enough when $|a - \overset{x^2}{b}| > \varepsilon * \overset{x^2}{\max(a, b)}$

If a and b are positive, and $\varepsilon = 10^{-9}$

abs(a – b) > 1e-9 * max(a, b)

Practice: find $\log_{10}$ a using bisection method

Write a program to get *a* as input and find the $\log_{10} a$ using bisection method, where $1 \le a \le 600$

To be more challenge, try *a,* which is much larger than 600

for loop

```
for k in range(start, stop, step) :
    …
```

```
for c in a_string :
    …
```

```
for e in a_list :
    …
```

Form #1:
for k in range(start, stop, step)

```
for k in range(4, 100, 2) :
    …
```

k = 4, 6, 8, …, 98

```
for k in range(100, 0, -1) :
    …
```

k = 100, 99, 98, …, 1

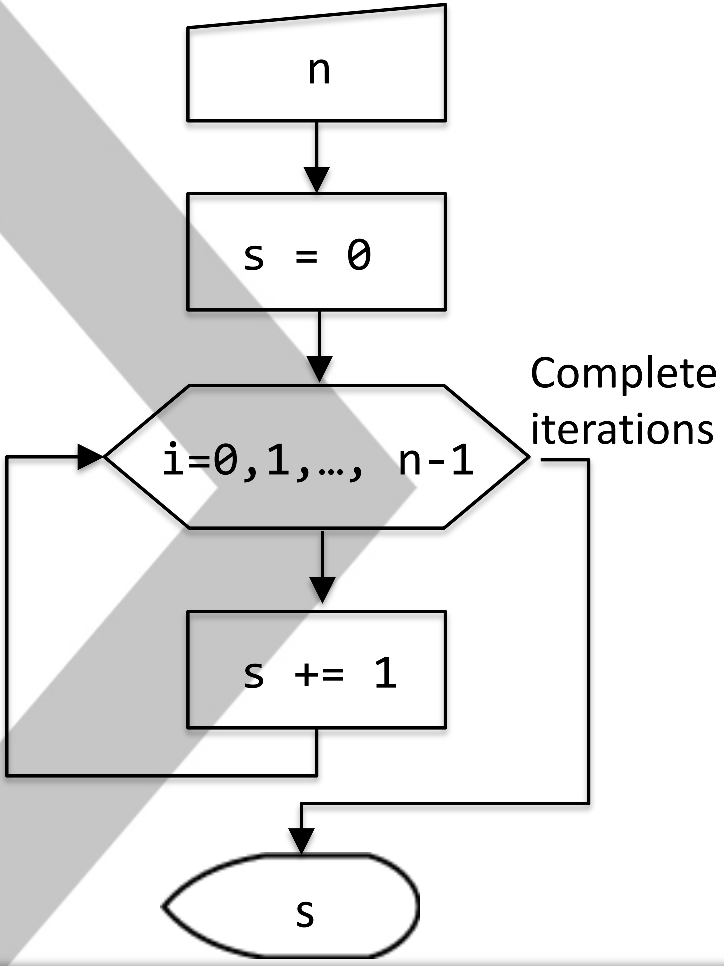
```
for k in range(5, 100) :
    …
```

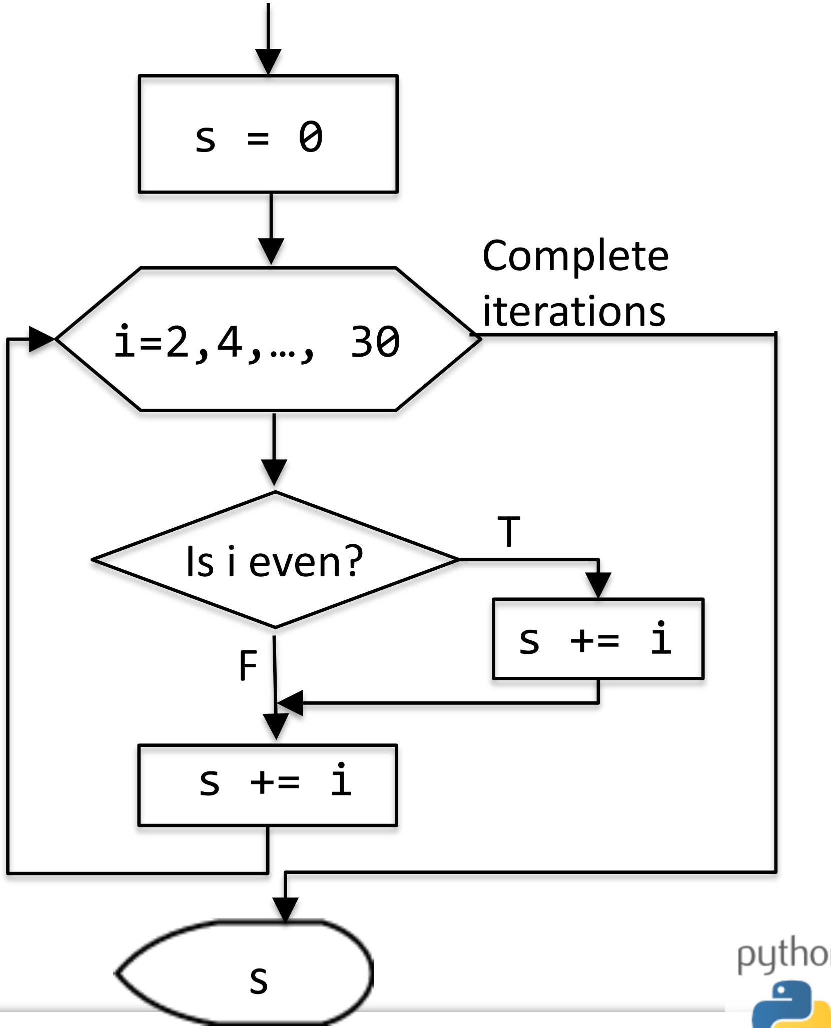
k = 5, 6, 7, …, 99

```
for k in range(100) :
    …
```

k = 0, 1, 2, …, 99

Form #1:
for k in range(start, stop, step)





2+2+4+4+6+6+8+8+…+30+30

Differences between while and for

```
s = 0
k = 0
while k < 5:
    s += (2*k – 1)**2
    k += 1

print(s)
```

```
s = 0

for k in range(0,5,1):
    s += (2*k – 1)**2


print(s)
```

```
min_v = float(input())
k = 0
while k < 4:
    v = float(input())
    if v < min_v:
        min_v = v
    k += 1
print(“min_v”, min_v)
```

```
min_v = float(input())

for k in range(4):
    v = float(input())
    if v < min_v:
        min_v = v

print(“min_v”, min_v)
```

Example: find μ and σ

```
N = int(input())
x = [0.0]*N
for i in range(N):
    x[i] = float(input())

s = 0
for i in range(N):
    s += x[i]
mean = s/N

s2 = 0
for i in range(N):
    s2 += (x[i]-mean)**2
sd = (s2/N)**0.5

print(mean, sd)
```

Input

```
5
10
11
13
10
12
```

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{N}}$$

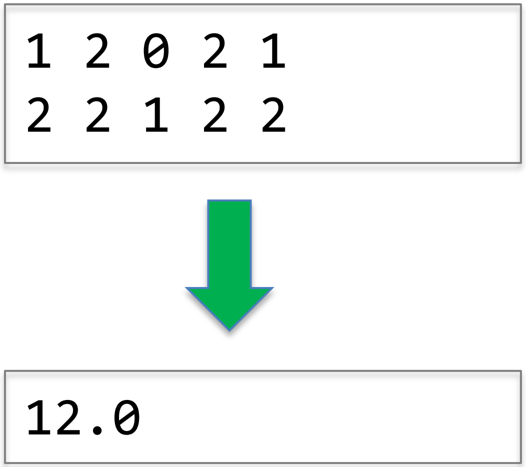
Example: Dot Product u·v

```
x = input().split()
u = [0.0]*len(x)
for i in range(len(x)):
    u[i] = float(x[i])

x = input().split()
v = [0.0]*len(x)
for i in range(len(x)):
    v[i] = float(x[i])

dot = 0
for i in range(len(x)):
    dot += u[i]*v[i]

print(dot)
```



```
1 2 0 2 1
2 2 1 2 2
```

```
12.0
```

Practice: check if the answer is correct

```
sol = input()   # e.g., ABBBAAABCCBABABDCCDA
ans = input()   # e.g., ABBBAABBCCBABABDCDDB


# count the number of matched characters between sol
# and ans
```

Form #2:
for each_character in a_string

Example: count number of digits in a string

```
s = input()
digit_counts = 0
for ch in s:
    if "0" <= ch <= "9":
        digit_counts += 1
print(digit_counts)
```

Which one is easier?

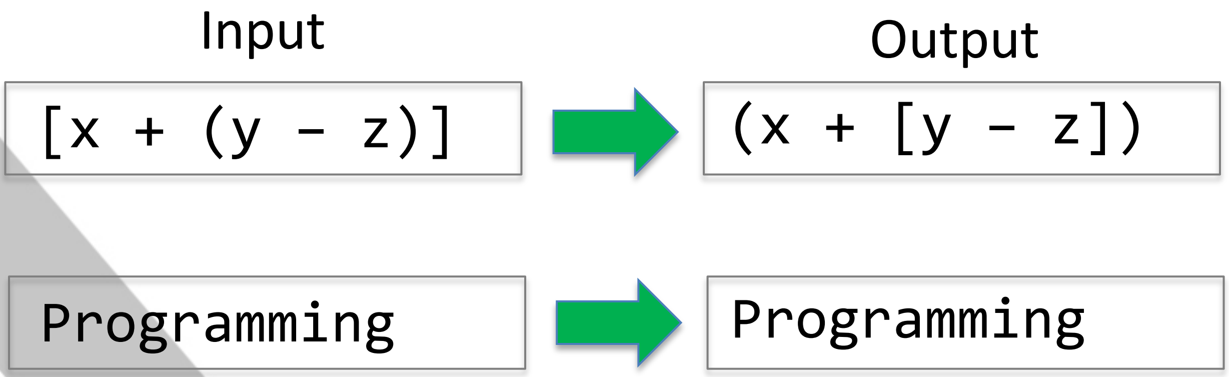
```
s = input()
digit_counts = 0
for i in range(len(s)):
    if "0" <= s[i] <= "9":
        digit_counts += 1
print(digit_counts)
```

Example: remove all ([{ }]) from an input string

```
s = input()
result = ""
for ch in s:
    if ch not in "([{}])":
        result += ch
print(result)
```

Check character by character, if it is not any of the ([{ }]), concatenate that character to the result variable for the output.

Practice: [ ] and ( )



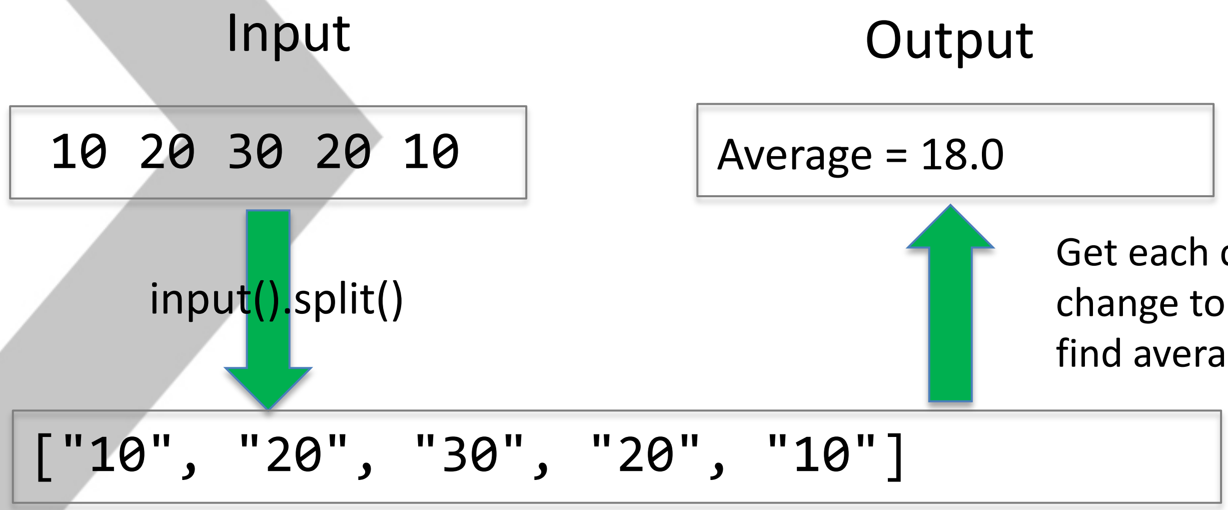
Input	Output
`[x + (y - z)]`	`(x + [y - z])`
`Programming`	`Programming`

Construct a new string, replace ( ) with [ ] and vice versa.

Form #3:
for each_element in a_list

Example: find an average

```
x = input().split()
s = 0
for e in x:
    s += float(e)
avg = s/len(x)
print("Average = ", avg)
```



Input

```
10 20 30 20 10
```

input().split()

```
["10", "20", "30", "20", "10"]
```

Output

```
Average = 18.0
```

Get each character one by one, change to float, add to sum, and find average.

Tips: Form#3 is applicable with both string and list

Get one by one character from *left to right.*

```
for e in x:
    …
```

Get one by one character from *right to left.*

```
for e in x[::-1]:
    …
```

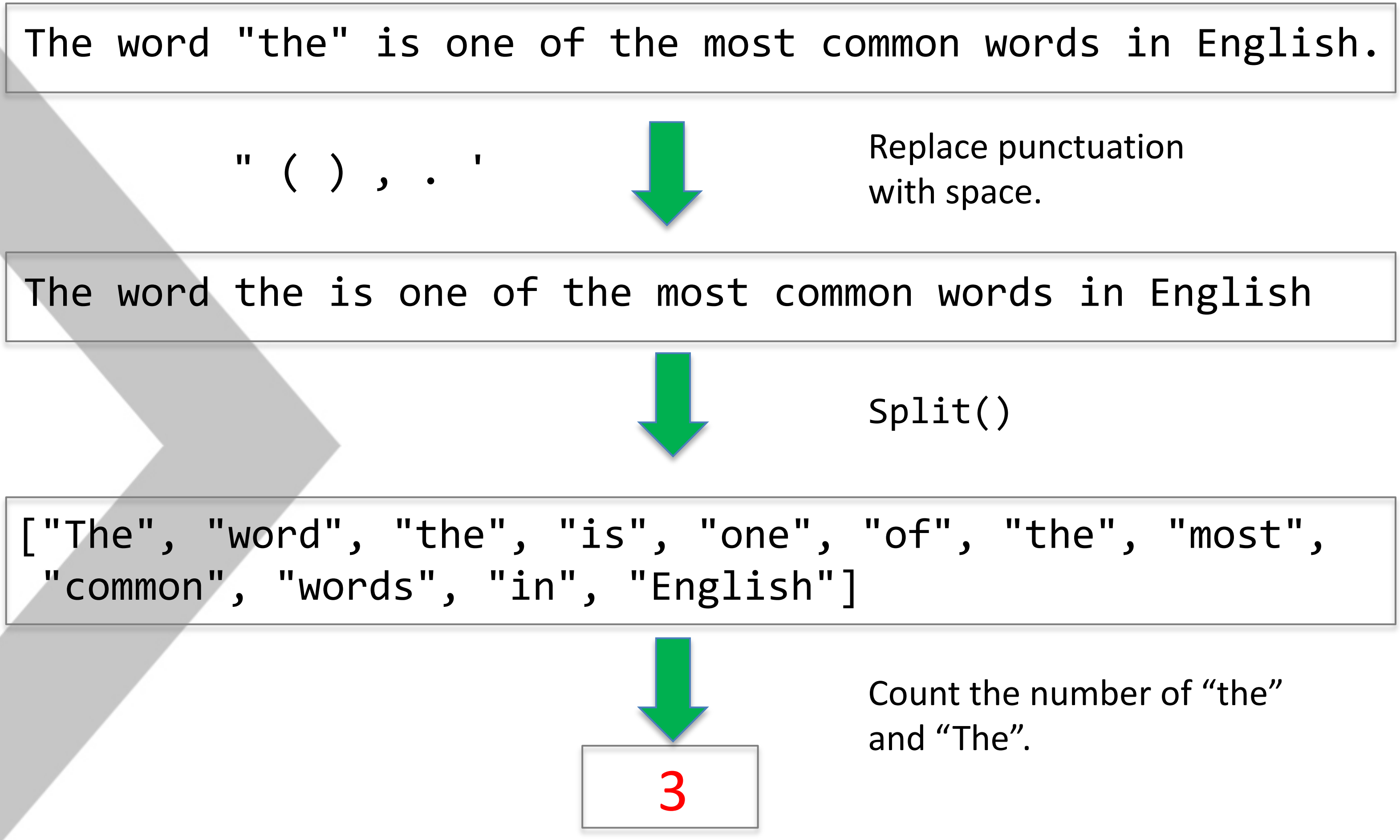
Get one by one character from *left to right* but not include the last character.

```
for e in x[:-1]:
    …
```

Get only characters *with odd index* from *left to right.*

```
for e in x[1::2]:
    …
```

Practice: count the number of "the" and "The"



```
The word "the" is one of the most common words in English.
```

```
" ( ) , . '
```

Replace punctuation with space.

```
The word the is one of the most common words in English
```

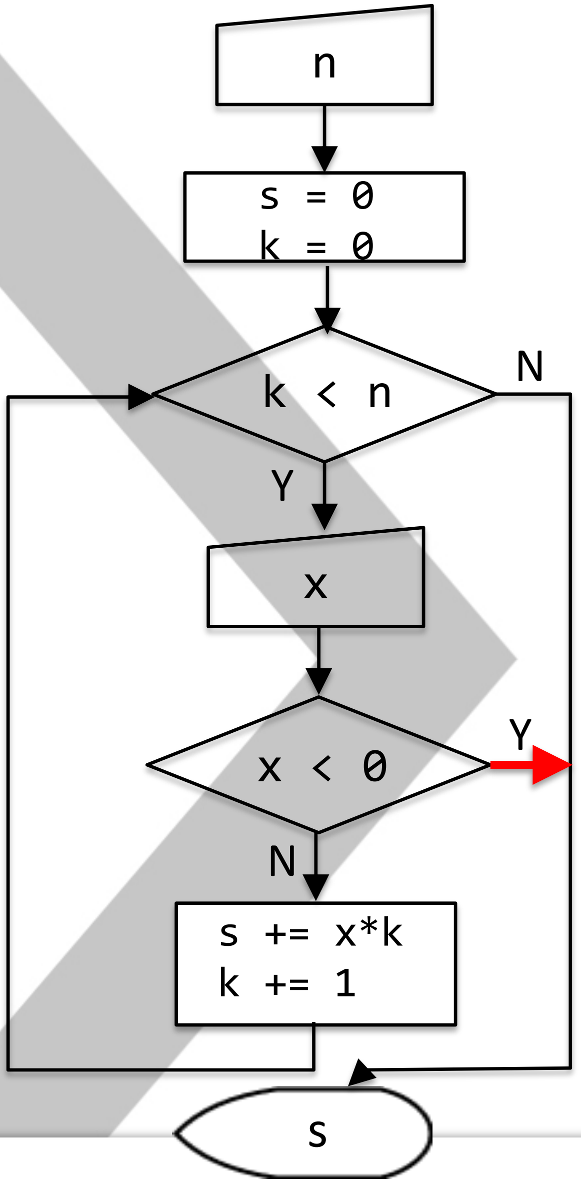
```
Split()
```

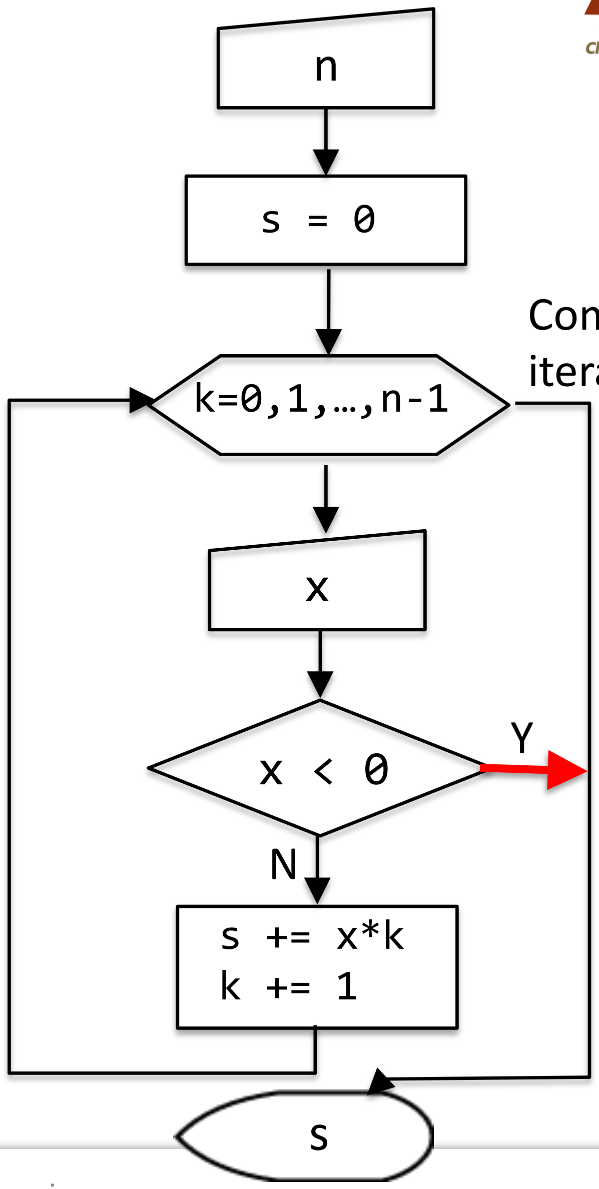
```
["The", "word", "the", "is", "one", "of", "the", "most",
 "common", "words", "in", "English"]
```

Count the number of “the” and “The”.

3

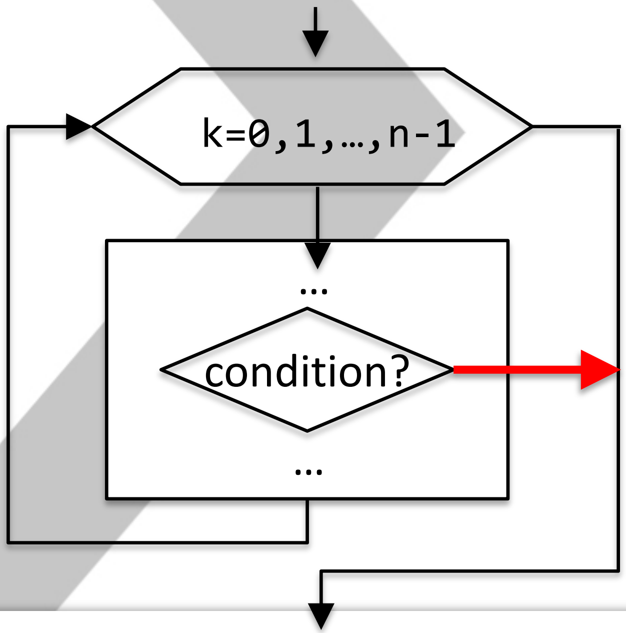
break: instruction to get out of a loop





break: instruction to get out of a *for* loop

```
for k in range(n):
    …
    if condition:
        break
    …
```



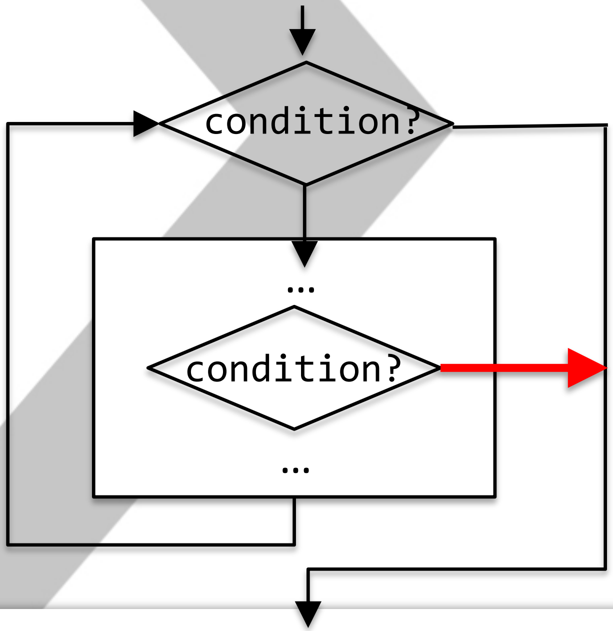
```
n = int(input())
for k in range(2, n+1):
    if n%k == 0:
        break
if k == n:
    print("Prime")
else:
    print("Composite")
```

Get *n* from keyboard. Find all *k* that are *n* divisible, k = 2, 3, …, n

When get out of the loop,
if k is equal to n, n is a prime number,
otherwise, n is a composite number.

break: instruction to get out of a *while* loop

```
while condition:
    …
    if condition:
        break
    …
```



```
t = input()
s = 0; n = 0
while t != "q":
    s += float(t)
    n += 1
    t = input()
print("Average =", s/n)
```

```
10.0
10.0
11.0
12.0
q
```

```
s = 0; n = 0
while True:
    t = input()
    if t == "q":
        break
    s += float(t)
    n += 1
print("Average =", s/n)
```

```
10.0
10.0
11.0
12.0
q
```

Practice: guess number game

Output

```
Guess my number (0 to 99)
You have seven tries
50
Higher
75
Lower
57
Higher
68
Lower
62
Lower
60
Lower
69
You win
```

Output

```
Guess my number (0 to 99)
You have seven tries
1
Higher
2
Higher
3
Higher
4
Higher
5
Higher
6
Higher
7
Higher
You lose, the number is 9
```

Practice: guess number game

```
import random

print("Guess my number (0 to 99)")
print("You have seven tries")
n = random.randint(0,99)
for k in range(7):
    m = int(input())
```

Randomly select a number between 0 to 99, inclusive

Have only 7 tries